FPGA-Based Filterbank Implementation for Parallel Digital Signal Processing [1]

Stephan Berner and Phillip De Leon
New Mexico State University
Center for Space Telecommunications and Telemetering
Box 30001, Dept. 3-0
Las Cruces, New Mexico 88003-8001
{sberner, pdeleon}@nmsu.edu

Abstract – **One approach to parallel digital signal processing decomposes a high bandwidth signal into multiple lower bandwidth (rate) signals by an analysis bank. After processing, the subband signals are recombined into a fullband output signal by a synthesis bank. This paper describes an implementation of the analysis and synthesis banks using FPGAs.**

1 Introduction

In the last decade research in filter banks and subband processing as well as multi-resolution analysis (MRA) a.k.a. wavelet transforms has occurred at a fast pace [12],[14],[6],[3]. In particular, subband processing of signals is now common in a number of applications such as audio, image, and video encoders/decoders; acoustic echo cancelers used in hands-free teleconferencing systems; and spread spectrum communications systems [1]. In addition, emerging applications include multiple target tracking in radar, high-data rate modems for satellite channels, and suppression of interference signals in wireless applications [8],[13],[2]. As compared to equivalent fullband processing, subband signal processing in these applications often leads to a performance increase due to the MRA and potentially a reduction in computation due to the lower sampling rate of the subband (component) signals. Another emerging area of application is in parallel digital signal processing architectures [5].

There are essentially two approaches in which to utilize multiple processors in an architecture. 1) The first approach employs program partitioning which divides the various tasks of an algorithm and assigns a processor to each task. This approach relies on algorithm transformation techniques (program unfolding, retiming, index mapping, and look-ahead transforms) in determining the partitioning of the program [11]. While this approach is well-suited for some applications, issues such as load balancing and program routing must also be addressed which often places a significant burden on the developer and requires sophisticated compiler and operating systems as well as support for inter-processor communications for data transport [9]. In addition, this approach only scales to a certain point before adding more processors leads to no computational gain. 2) The second approach employs data decomposition, which divides the data and assigns a processor a subset of the data. This approach has also proved successful in a number of applications including search problems

[1]This research was supported by NASA under Research Grant NAG 5-7520.

where each processor can search a different path in the space, however, this approach typically breaks down with problems involving times-series data that must computed sequentially such as trajectories. In this case the calculation of the position at time, t requires calculation of the position at time $t - \Delta$, and so on. However, this approach scales well in that we need only produce finer data decompositions to utilize additional processors.

While the first approach is often used in parallel signal processing, it is possible to employ the second approach despite the sequential nature of the time-series data (samples of the signal) to reduce the burdens of the first approach while at the same time, enjoying the scalability advantage. In this approach, a subband decomposition (filter bank) analyzes the signal into independent components. The independent components are then processed concurrently with the outputs synthesized to form the result. One very attractive feature of the subband approach to parallel signal processing is that the processing software of each signal component can often be made identical and performed in lock-step (as in the search problem) thus bypassing many issues inherent in the first approach.

Preliminary work into the use of subband decompositions for parallel signal processing is being investigated at NMSU's Center for Space Telemtering and Telecommunications for application to high-data rate digital receivers [5]. This paper reports on an FPGA-based implementation of a high-speed polyphase, uniform-DFT filter bank which is to be used in a parallel signal processing architecture. We begin with a short review of filter bank theory, a description of the implementation, and test results and data from the implementation.

2 Filter Bank Review

2.1 Uniform-DFT Filter Banks

An analysis bank decomposes a signal, x into frequency bands or subbands by using a set of M bandpass filters (BPFs) denoted $\mathbf{f}_0, ..., \mathbf{f}_{M-1}$ in Figure 1. Once partitioned, the signals are then downsampled by a factor of D (due to their reduced bandwidth). Upon analysis, each of these subband signals $(x_0, ..., x_{M-1})$ may be processed (subband processing) independently of the others. After processing, the subband signals are synthesized into the fullband signal by first upsampling (zero insertion) to the original sampling rate followed by bandpass filtering $(\mathbf{g}_0, ..., \mathbf{g}_{M-1})$ to remove spectral images introduced from upsampling. Such a system of upsamplers and bandpass (synthesis) filters is referred to as a synthesis bank. The combination of analysis and synthesis banks is often called a filter bank [14]. If the filter bank is designed properly, the parallel processing of the subband signals along with proper synthesis can often be made equivalent to fullband processing [14].

One of the more popular designs for filter banks, called the uniform-DFT filter bank, assumes that all analysis, synthesis filters are frequency-shifted versions of an analysis, synthesis lowpass filter (LPF) prototype, $\mathbf{f}_0$, $\mathbf{g}_0$ respectively. The time-domain relations are given by

$$\begin{aligned} \mathbf{f}_m(n) &= \mathbf{f}_0(n)e^{j2\pi nm/M} \\ \mathbf{g}_m(n) &= \mathbf{g}_0(n)e^{j2\pi nm/M} \end{aligned} \tag{1}$$

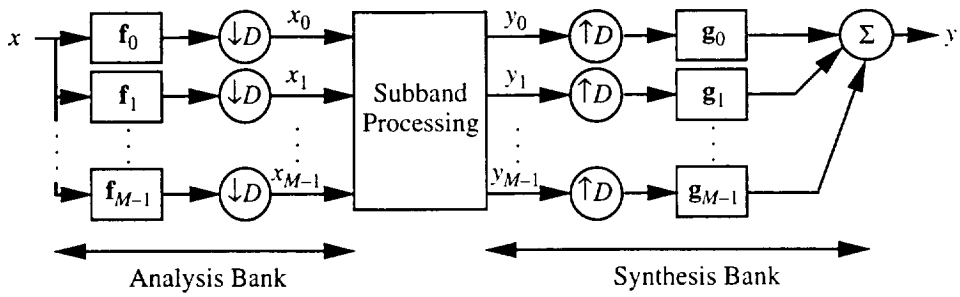

Figure 1: *M-subband, filter bank with subband processing*

for $m = 1, \ldots, M-1$ and the frequency-domain relations are given by

$$\begin{aligned} F_m\left(e^{j\omega}\right) &= F_0\left(e^{j(\omega - 2\pi m/M)}\right) \\ G_m\left(e^{j\omega}\right) &= G_0\left(e^{j(\omega - 2\pi m/M)}\right). \end{aligned} \tag{2}$$

A typical magnitude response of analysis/synthesis filters related by (2) with $M = 4$ is given in Figure 2.

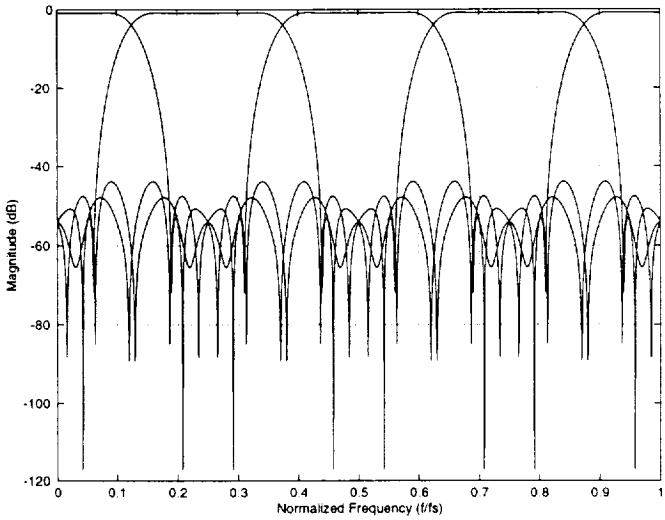

Figure 2: *Magnitude responses of analysis/synthesis filters of a uniform-DFT filter bank*

2.2 Polyphase, Uniform-DFT Filter Banks

In the analysis bank shown in Figure 1, much computation is wasted in the implementation since after filtering, only one out of every D samples is retained. Similar filtering inefficiencies occur in the synthesis bank since only one out of every D samples is non-zero prior to synthesis filtering. A more efficient but equivalent implementation, called a polyphase, uniform DFT filter bank relies on a polyphase representation of $\mathbf{f}_0$ and $\mathbf{g}_0$ (LPF prototypes) and a discrete Fourier transform (DFT) and inverse DFT (IDFT) as shown in Figure 3 [12]. Note that in Figure 3, $I = M/D$ is referred to as the oversampling factor and will be discussed in

the next section. The polyphase representation of the prototype LPFs used in Figure 3 is given by

$$\begin{aligned} \mathbf{a}_m(n) &= \mathbf{f}_0(nD-m) \\ \mathbf{r}_m(n) &= \mathbf{g}_0(nD-m) \end{aligned} \tag{3}$$

where $m = 0, \ldots, M-1$. As an example, if we assume $M = 4, D = 2$, and a causal, FIR analysis filter prototype, $\mathbf{f}_0$ the polyphase filters would be given by

$$\begin{aligned} \mathbf{a}_0(n) &= [\mathbf{f}_0(0), \mathbf{f}_0(2), \mathbf{f}_0(4), \mathbf{f}_0(6), \ldots]^T \\ \mathbf{a}_1(n) &= [0, \mathbf{f}_0(1), \mathbf{f}_0(3), \mathbf{f}_0(5), \ldots]^T \\ \mathbf{a}_2(n) &= [0, \mathbf{f}_0(0), \mathbf{f}_0(2), \mathbf{f}_0(4) \ldots]^T \\ \mathbf{a}_3(n) &= [0, 0, \mathbf{f}_0(1), \mathbf{f}_0(3), \ldots]^T . \end{aligned} \tag{4}$$

With the polyphase representation, analysis and synthesis filtering occurs at the lower, subband sampling rate resulting in a lower processing speed. In addition, the DFT and IDFT are implemented with a fast-Fourier transform (FFT). These combined effects result in a significant computational reduction. For the implementation of the polyphase, uniform-DFT filter

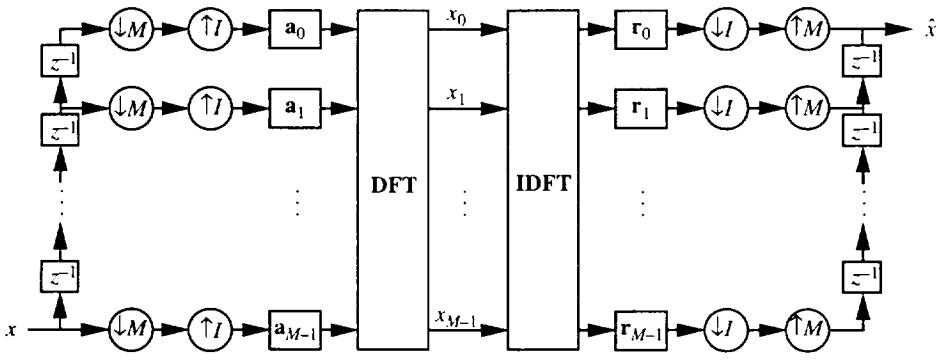

Figure 3: *Polyphase, uniform DFT filter bank (I-times oversampled)*

bank, it is convenient to view the polyphase structures in terms of a commutator model as illustrated in Figure 4 [12]. In the analysis stage, the commutator rotates counter-clockwise beginning with the Mth subband at time $n = 0$. The commutator is thus equivalent to a delay chain feeding a bank of downsamplers. In the synthesis stage, the commutator rotates clockwise beginning with the 0th subband at time $n = 0$.

2.3 Oversampled, Polyphase, Uniform-DFT Filter Banks

In most of the literature, filter bank design is aimed at perfectly reconstructing x (to within a scale factor and delay) from the critically downsampled ($D = M$ or $I = 1$) subband signals $x_0, \ldots, x_{M-1}$ (Figure 4). Critically downsampled refers to the fact that the number of fullband samples per second equals the total number of subband samples per second. In this case, one must carefully control the aliasing in the subband signals through proper design of the analysis and synthesis filters. Many critically downsampled designs are available which

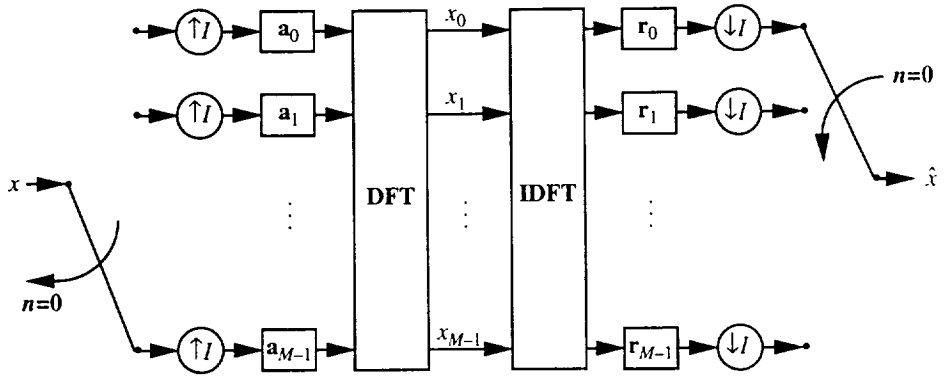

Figure 4: *Commutator model of the polyphase, uniform-DFT filter bank (I-times oversampled)*

yield the perfect reconstruction property. The fundamental problem in processing critically sampled subband signals, however, is the large level of aliasing which is present regardless of the analysis and synthesis filter designs due to the overlapping filters. With such a large amount of aliasing (which acts as a disruptive noise signal), many subband signal processing applications simply cannot be made equivalent to their fullband counterparts or they require a large additional overhead to partially compensate for the aliasing [7].

The solution to the subband aliasing problem is to oversample ($D < M$) the subband signals to minimize the effect of the aliasing [12] [4]. We typically choose the oversampling factor $I = 2$ for implementation reasons, which implies twice the total number of samples per second in the subbands than the fullband rate. Even with oversampled subbands, many signal processing applications can have fewer total computations per sample as compared to the fullband case. In any case, with minimal subband aliasing, parallel processing of the subbands can now be made equivalent to fullband processing (with delay).

3 Implementation using FPGAs

The oversampled, polyphase, uniform-DFT filter bank is implemented on a pair of Field Programmable Gate Arrays (FPGAs). We use one FPGA for the analysis bank and another for the synthesis bank. In the filterbank implementation, all components of the filter bank are described in Very high speed integrated circuit Hardware Description Language (VHDL) because of the flexibility VHDL offers. This flexibility can be used for purposes of scaling the architecture (adding more subbands/processors), increasing wordsizes, and/or using sharper analysis/synthesis filters. For example, changing the number of subbands or the wordsize is easily done in the description by altering a few constants in the source file. A filter bank with four, eight, and sixteen subbands has been described in VHDL and simulated. In addition, the four subband filter bank has been implemented, tested, and verified on a pair of Xilinx XC4025 FPGAs [15]. In this section we describe implementation of the various key components of the filter bank.

3.1 Commutator

Figure 5 illustrates the digital implementation of the commutator for the filter bank. The shift register includes only one "one" and the other bits are cleared. The "one" rotates around the shift register enabling one data register at a time with every rising clock edge which is then loaded with the current sample, i.e. a simple "one hot" state machine. The shift register is initialized with the binary value `%0...01`. In the VHDL description of the commutator, we employ a loop that builds as many elements as needed depending on the desired number of subbands.

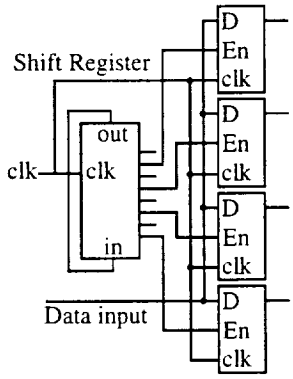

Figure 5: *Commutator implementation for four subband filter bank*

3.2 Upsampler

Figure 6 illustrates the digital implementation of the upsampler for the filter bank used in the oversampling process as illustrated in Figure 4. The implementation employs a simple multiplexer which inserts a "zero" every other sample in order to achieve $2\times$ oversampled subbands.

3.3 Analysis/Synthesis Filters

The polyphase filters are derived as in (3) from the analysis and synthesis prototype LPFs. We design a length-32 prototype filter using the Parks-McClellan linear phase filter design algorithm (also known as "remez" in MATLAB). In the design, the filter coefficients are normalized to an 8-bit representation. The overall impulse response of the filter bank is shown in Figure 7 where we see aliasing artifacts no more than 40dB below the main impulse. We note that since the filter bank is not time invariant but rather periodically time-varying, we show the worst-case (largest aliasing artifact) impulse response.

Once the design is complete, we use the Xilinx Core Generator to generate the filters. For efficient use of hardware resources we use Serial Distributed Arithmetic (SDA) in the calculation of filter output although we may optionally employ Parallel Distributed Arithmetic (PDA) for higher speed (at the expense of additional area on the array). With a length-32 polyphase filter (8-bit coefficient size) we are able to achieve a maximum subband input rate of 8.9MHz. In the actual implementation we choose a subband input rate of 8MHz

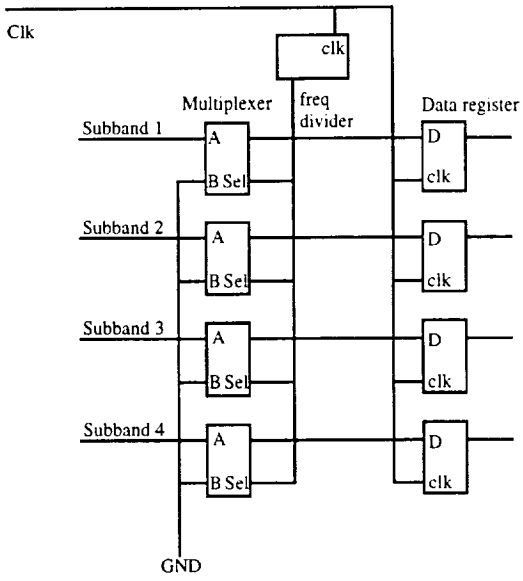

Figure 6: *Upsampler implementation for twice oversample subbands*

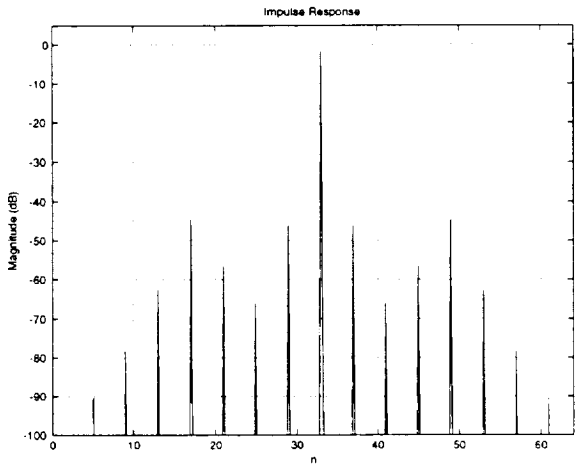

Figure 7: *Worst case impulse response for filter bank*

for convenience. This leads to different fullband input rates depending on the number of subbands as in Table 1.

3.4 FFT/IFFT

In the literature there are many different methods given to implement an FFT, ranging from fully parallel (high speed, high area consumption) to fully serial (low speed, low area consumption) [10]. For this project a "pipeline FFT" was selected because it provides only as much parallelism as necessary, and therefore does not consume too much area. Furthermore, it is easily scaleable for different numbers of subbands. The "pipeline FFT" makes use of the fact that as soon as butterfly (basic computational unit in the FFT) outputs in the previous stage are calculated, their results are immediately used in the calculation of butterfly outputs

Table 1: Datarates/word sizes in filter bank

	Number of Subbands, M		
	4	8	16
Input Signal	16MHz/8-bit	32MHz/8-bit	64MHz/8-bit
Commutator Output	4MHz/8-bit	4MHz/8-bit	4MHz/8-bit
Upsampler Output	8MHz/8-bit	8MHz/8-bit	8MHz/8-bit
Filter Output	8MHz/21-bit	8MHz/21-bit	8MHz/21-bit
FFT Output	8MHz/22-bit	8MHz/23-bit	8MHz/24-bit
Analyzer Output	8MHz/23-bit	8MHz/24-bit	8MHz/25-bit
Synthesizer Input	8MHz/8-bit	8MHz/8-bit	8MHz/8-bit
IFFT Output	8MHz/10-bit	8MHz/11-bit	8MHz/12-bit
Filter Output	8MHz/21-bit	8MHz/21-bit	8MHz/21-bit
Downsampler Output	4MHz/21-bit	4MHz/21-bit	4MHz/21-bit
Commutator Output	16MHz/21-bit	32MHz/21-bit	64MHz/21-bit
Output Signal	16MHz/21-bit	32MHz/21-bit	64MHz/21-bit

of the next stage (Figure 8).

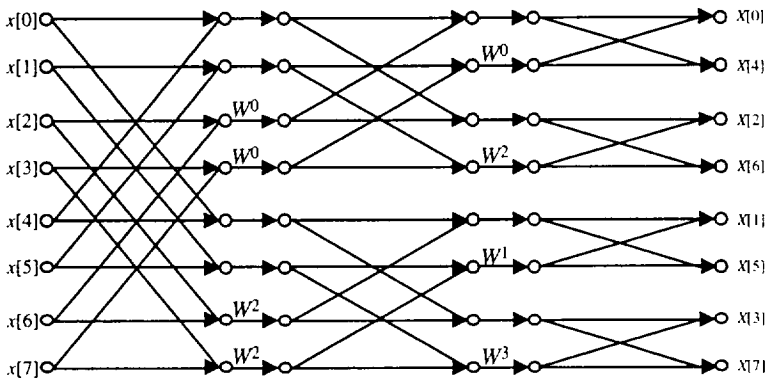

Figure 8: *FFT flowgraph*

The design of the pipeline FFT is illustrated in Figure 9. Data from the polyphase analysis filters is written (at an input rate of 8Mhz) into the first (parallel in, serial out) shift registers. The output of the shift registers (serial) is moved into the pipeline at a rate of 64Mhz (which is the system clock). From the flow graph above it can be seen that after the calculation of the first two butterflies of the first stage the cross switch has to be toggled, and a multiplication of the lower butterfly output by j has to be performed. This operation is controlled by the delayed value of a control counter. Again we note the pipeline FFT is easily scaled for different number of subbands by adding additional pipeline stages.

3.5 Array Consumption

The area consumed on the FPGA naturally increases with more subbands. Listed in Table 2 are the number of Configurable Logic Blocks (CLBs) (basic unit in the Xilinx FPGA)

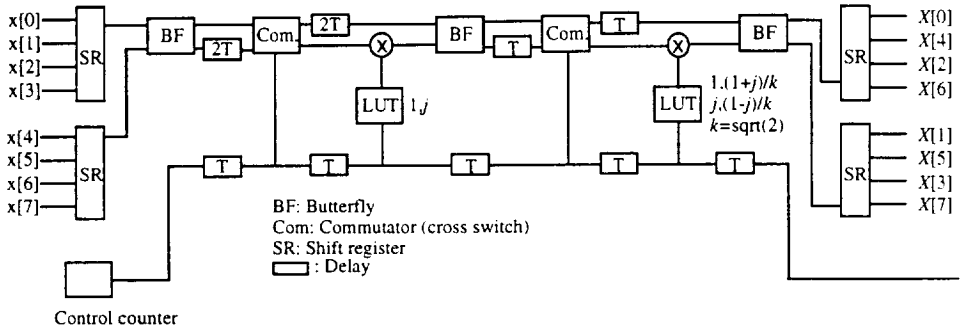

Figure 9: *Pipeline FFT*

Table 2: Required FPGA CLBs for implementation of the filter bank

	Number of Subbands, M		
	4	8	16
Analyzer w/o filters	273CLBs	688	1525
Polyphase Analysis Filters (length 32)	428	856	1712
Analysis Bank (Total)	701	1544	3237
Synthesizer w/o filters	208	463	913
Polyphase Synthesis Filters (length 32)	428	856	1712
Synthesis Bank (Total)	680	1407	2861

required for implementation. We note the basic Xilinx XC4025E device contains 1024CLBs. Other devices in the family are available which have more CLBs.

4 Test Results

Instead of using the ADC, the carry signal of an 8-bit counter, which is clocked with the sample clock, is used to generate an input impulse train. The space between pulses (255 "zeros") is big enough to allow the response to decay completely before the next pulse is applied, the space is small enough, however, to display the response on an oscilloscope. The impulse response was captured with a digital scope and values were verified against the MATLAB simulation in Figure 7.

5 Conclusion

In this paper we have described an FPGA-based implementation of a filter bank. The filter bank is capable of operating at input sampling rates of 16MHz (4 subbands), 32MHz (8 subbands), and 64MHz (16 subbands) and forms the basis for a parallel digital signal processing architecture which takes advantage of multiple, lower-rate subband signals.

References

[1] A. Akansu and M. Smith, editors, *Subband and Wavelet Transforms: Design and Applications*, Kluwer Academic Publishers, Norwell, MA, 1996.

[2] A. Akansu, M. Tazebay, M. Medley, and P. Das, "Wavelet and Subband Transforms: Fundamentals and Communication Applications," *IEEE Communications Magazine*. p.104-115, Dec. 1997.

[3] C Burrus, R. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms*. Prentice-Hall, Upper-Saddle River, NJ, 1998.

[4] Z. Cvetkovic and M. Vetterli, "Oversampled Filter Banks," *IEEE Transactions on Signal Processing*, vol. 46, p. 1245-1255, May 1998.

[5] P. De Leon, "On the Use of Filter Banks for Parallel Digital Signal Processing," *7th NASA Symposium on VLSI Design*, (Albuquerque, NM), 1998.

[6] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[7] A. Gilloire and M. Vetterli, "Adaptive Filtering in Subbands with Critical Sampling: Analysis, Experiments, and Applications to Acoustic Echo Cancellation." *IEEE Transactions on Signal Processing*, vol. 40, p. 1862-1875, Aug. 1992.

[8] L. Hong, "Multiresolutional Multiple-Model Target Tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, p. 518-524, Apr. 1994.

[9] K. Hwang and Z. Xu, *Scalable Parallel Computing*, McGraw-Hill, Boston, MA, 1998.

[10] B. Liu, *Digital filters and the Fast Fourier Transform*, Dowden, Hutchinson & Ross, Inc., Stroudsburg, PA, 1975.

[11] K. Parhi, "Algorithm Transformation Techniques for Concurrent Processors," *Proceedings of the IEEE*, vol. 77, no. 12, p. 1879-1895. Dec. 1989

[12] L. Rabiner and R. Crochier, *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[13] R. Sadr, P. P. Vaidyanathan, D. Raphaeli, and S. Hinedi, "Parallel Digital Modem using Multirate Digital Filter Banks," *JPL Publication*, 94-20, Aug. 1994.

[14] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[15] www.xilinx.com 1999